

Vision 2000 Control Center System

System Monitoring

Manage Events

Release 2.1

Design Walk-through

Contents

| | |
|----------------------------------------------------------|-----------|
| CONTENTS..... | 2 |
| 1.0 HIGH LEVEL FUNCTIONAL DESCRIPTION..... | 3 |
| 1.1 LIFE CYCLE OF AN EVENT MESSAGE | 3 |
| 1.2 DEFINITIONS AND EXAMPLES..... | 4 |
| 2.0 DATA FLOW DIAGRAMS..... | 6 |
| 3.0 PROCESS DESCRIPTIONS..... | 7 |
| 3.1 APPLICATION PROGRAMMING INTERFACE..... | 7 |
| 3.2 MANAGE EVENTS (TDA DFD 3.6) | 7 |
| 3.3 RELATED REQUIREMENTS..... | 8 |
| 4.0 INTERFACES..... | 11 |
| 5.0 OBJECT MODELS & EVENT TRACE DIAGRAMS..... | 14 |
| 6.0 CLASS DESCRIPTIONS..... | 15 |
| 6.1 ATTRIBUTES AND METHODS | 15 |
| 7.0 FUTURE CAPABILITIES..... | 23 |

List of Tables

| | |
|----------------------------------------------------------|----|
| TABLE 1 EVENT LOOKUP & EVENT LOG | 4 |
| TABLE 2 VISION 2000 CCS SYM EVT INTERFACES..... | 11 |
| TABLE 3 EVENT MESSAGE SIZE | 20 |
| TABLE 4 EVENT MESSAGE SOURCE_SUBSYSTEM ENUMERATION | 20 |
| TABLE 5 EVENT MESSAGE TYPE ENUMERATION | 21 |
| TABLE 6 EVENT MESSAGE SEVERITY ENUMERATION..... | 21 |

1.0 High Level Functional Description

This section describes the Manage Events (EVT) functionality for Vision 2000 Control Center System (CCS) Release 2.1. The EVT process will aid in development and integration testing and support health and safety analysis during spacecraft operations. EVT will be the controlling process for all event messages created within the CCS. This will allow for enhanced maintainability and increased modularity for future extensions or changes to the system.

The EVT will perform the following functionality for release 2.1:

- Provide an Application Programming Interface (API) for C++ applications within CCS to generate event messages. These event messages will then be routed to a central process for completion and distribution to other CCS processes.

Note: Event API will not support applications written in JAVA, Fortran, or C. Developers using these languages will have to write their own interface.

- The EVT will route real time event messages to the DMG subsystem for archiving.
- The EVT will route event messages to the GUI subsystem for display purposes.
- The EVT will finish building the event message object with type, severity, and background attributes.

Note: EVT will provide additional functionality for future releases. A description of future capabilities are listed in section 7.0.

The following items will have to be provided by external CCS teams to support release 2.1:

- DMG to provide an Event Lookup Table containing the following static information about each event message; i.e., ID Number, Severity, Type, and Background Text.
- DMG to provide a proxy class to query the Event Lookup Table.
- DMG to provide an archive capability for event messages.
- DMG to provide a proxy class to log event messages.
- GUI to provide a user interface to enable users to monitor real time events.
- Middleware to provide classes allowing event message objects to be passed around the network.

1.1 Life Cycle of an Event Message

- The Event API will have an EventGenerator object. The EventGenerator will allow CCS C++ applications to call a method to instantiate an event message Object.

Note: A full description of the Event API is provided in the Vision 2000 CCS System Monitor's Manage Events Application Programming Interface Developers Guide Release 2.1

Note: CCS_Time provided by the EventGenerator will ensure that event messages are time stamped as close as possible to the original time that they are created. This method could result in "out of sequence" events at the central location and at SYM EVT clients such as GUI and CCM. The CCS development approach is to sort event messages at the client side (i.e. GUI, CCM, DMG, and SYM) of events if needed.

- The event message object will be routed to the EVT for further processing.

Questions or comments please contact Douglass George in Rm 35

E-mail: dgeorge@v2kmail.gsfc.nasa.gov

Phone: 918-7488

Together, foreground and background data are merged to create an ASCII text message.

Example:

User Doug George requested 6 analysis plots.

Questions or comments please contact Douglass George in Rm 35

E-mail: dgeorge@v2kmail.gsfc.nasa.gov

Phone: 918-7488

2.0 Data Flow Diagrams

The System Monitoring Level 2 DFD (3.0) DFD is attached at the end of this document.

3.0 Process Descriptions

This section provides a brief description of the Manage Events (3.6) process displayed in the CCS System Monitoring Level 2 DFD (3.0).

3.1 Application Programming Interface

Residing on every C++ application is the Event API. The EVT API provides the mechanism for CCS applications to generate and send event messages to the EVT process. The following subsystems will be allowed to generate and send event messages to the EVT:

- FEP
- Command Processing
- System Monitoring
- Data Management
- CCS Management
- GUI

Note: Middleware (MDW) is not considered a CCS subsystem but will have the capability to generate and send event messages to the EVT.

The FEP subsystem will generate event messages regarding the routing, display, and interpretation of downlink data concerning ODM/UPD and spacecraft telemetry. Uplink data concerning commands and table uploads will also be monitored within the FEP; corresponding event messages will be generated.

The CMD subsystem will generate event messages regarding command scheduling and processing. This includes receiving command requests, formatting, buffering, sending, and verifying spacecraft commands and Ground Control Message Requests (GCMR). Spacecraft orbital events such as ZOE and day/night transitions will also be generated by CMD.

The DMG subsystem will generate event messages regarding long term storage of CCS products. This includes items such as the Historical Integrated Command Schedule (HICS), telemetry, event log, event lookup table, and other products.

The CCM subsystem will generate event messages regarding CCS infrastructure and MUGSY configuration changes. This includes network status, HW status, SW status, time synchronization among CCS HW components, the IPC status among CCS components, and PDB changes.

The SYM system will generate event messages regarding the Control SYM, Distribute State Data, Detect Faults, Respond to Events & Faults, Perform Analysis and Trending, Perform Legacy, and Manage Monitor Data processes.

The GUI subsystem will generate event messages regarding the user interface software. This includes network and response message time outs, and user-related information such as login and logout activities.

3.2 Manage Events (TDA DFD 3.6)

This process finishes building an event message by completing the type, severity, background, and a string version of the process name. This information is obtained from the DMG subsystem. After completing an event message, EVT will distribute the message to subscribing clients. The GUI subsystem will receive all messages.

To prevent logging multiple copies of event messages, any redundant real time EVT process running on the Core or Backbone LANs will not send event messages to the DMG. In addition, event messages generated in historical mode will not be logged.

3.3 Related Requirements

The following requirements will be implemented in Release 2.1.

Note: Future release requirements are not listed, but may be found in section 7.0.

- 6.0.1** The EVT shall provide the capability for every application within the CCS to generate event messages.
- 6.0.2** The EVT shall provide CCS developers with the capability to inject event messages into the system for testing purposes.
- 6.0.3** The EVT shall allow CCS applications to log event messages as text for testing purposes.
- 6.1.1** Event messages shall have a subsystem source describing the CCS process that generated the event message.

Note: It is recommended that the following numbers identify the number ranges that are reserved for specific CCS subsystems.

- FEP
- CMD
- SYM
- DMG
- CCM
- GUI
- MDW

Note: Middleware (MDW) is not considered a CCS subsystem but will have the capability to access the Event API for generating event messages and sending event messages to the EVT.

- 6.1.2** Event messages shall have a node source describing the node of the application that generated the event message.

6.1.3 Event messages shall have a process name source describing the application that generated the event message.

6.1.4 Event messages shall have a process PID number describing the processes that generated the event message.

6.1.5 Event messages shall be categorized into one or more event types.

Note: Candidate Types include the following categories of event types:

- Telemetry Event
- Playback Event
- Real-time Event
- NCC Event
- Ground Configuration Event
- Alarm Event
- Command Dialog Event
- PSTOL Response Event
- General Event
- System or Software Error Event
- Keyboard Input Echo
- Command Page

6.1.6 Event messages shall have a time in UTC format to millisecond granularity describing the time that it was generated.

6.1.7 Event messages shall have unique identifier in integer format. The following lists the number ranges that are reserved for specific CCS subsystems.

Note: The following lists the recommended number ranges that are reserved for specific subsystems

- CCS 00,000 - 04,999
- MDW 05,000 - 09,999
- FEP 10,000 - 14,999
- CMD 20,000 - 24,999
- SYM 30,000 - 34,999
- DMG 40,000 - 44,999
- CCM 50,000 - 54,999
- GUI 60,000 - 64,999

Note: The 1st and 2nd ID numbers for each range are reserved. The 1st number (e.g., 10,000) is reserved for a general (no Background data) message. The 2nd number (e.g., 10,001) is reserved for a general debug message (no Background data & severity = debug).

6.1.8 Event Messages shall have a status indicating the operational mode of the application that requested the event message.

- 6.1.9** Event messages shall have a textual description that details the specific information the message conveys.
- 6.1.10** Event messages shall have a severity level of 0, 1, 2, or 3. Severity level 0 describes a debug event message. Severity level 1 describes a nominal operational without any urgency. Severity level 2 describes non-nominal situation with urgency. Severity level 3 describes an anomalous situation requiring immediate resolution.
- 6.1.11** The EVT shall have a status indicating if it is the single primary EVT process or a redundant Process
- 6.1.12** The EVT shall route all event messages to the DMG for archiving that meet the following criteria:
- From a real time application. The operational mode of the event messages is set to real time.
 - The EVT processing this event message is a primary process.

Note: Middleware will provide a mechanism for guaranteeing delivery of event messages to the DMG.

- 6.1.13** The EVT shall complete all event messages that are not of historical operational mode.
- 6.1.14** Future Requirement
- 6.1.15** Future Requirement
- 6.1.16** The EVT shall route all event messages to the GUI subsystem.

Note: Guaranteed delivery of event messages to the GUI is not required.

4.0 Interfaces

The following table lists the external interfaces with EVT as identified on the TDA Level 2 DFD and TDA Level 3 DFD.

Note: This table provides the interface descriptions for all releases.

Table 2 Vision 2000 CCS SYM EVT Interfaces

| Source | Level 2 Destination | Level 3 Destination | Level 2 Interface (Data Flow Name) | Level 3 Interface (Data Flow Name) | Description | Interface Elements | Mode |
|---------------------------------|------------------------|---------------------------------|---------------------------------------|---------------------------------------|-------------------------------------------------------------------------------------------------|------------------------|------------------------|
| FEP Interface | | | | | | | |
| FEP (1.0) | (3.6)SYM_ManageEvents | SYM_EvtHandleEvents (3.6.1) | G_FEP_Events | G_FEP_Events | Incomplete event message | • Event Message Object | Real Time |
| CMD Interface | | | | | | | |
| CMD (2.0) | (3.6)SYM_ManageEvents | SYM_EvtHandleEvents (3.6.1) | G_CMD_Events | G_CMD_Events | Incomplete event message | • Event Message Object | Real Time |
| SYM_CNT Interface | | | | | | | |
| ControlSYM (3.1) | (3.6)SYM_ManageEvents | SYM_EvtHandleEvents (3.6.1) | SYM_CntEvent | SYM_CntEvent | Incomplete event message | • Event Message Object | Real Time |
| ControlSYM (3.1) | (3.6)SYM_ManageEvents | SYM_EvtDistributeEvents (3.6.3) | SYM_CntReplayEvent | SYM_CntReplayEvent | Historical event message objects that are passed individually into EVT during Historical Modes. | • Event Message Object | Historical |
| SYM_EvtDistributeEvents (3.6.3) | ControlSYM (3.1) | TBD | SYM_EvtCntFilteredEvent | SYM_EvtCntFilteredEvent | Event Message Object | • Event Message Object | Real Time & Historical |
| SYM_DSD Interface | | | | | | | |
| SYM_DistributeStateData (3.2) | (3.6)SYM_ManageEvents | SYM_EvtHandleEvents (3.6.1) | SYM_DsdEvent | SYM_DsdEvent | Incomplete event message | • Event Message Object | Real Time |

Questions or comments please contact Douglass George in Rm 35

E-mail: dgeorge@v2kmail.gsfc.nasa.gov

Phone: 918-7488

| Source | Level 2 Destination | Level 3 Destination | Level 2 Interface (Data Flow Name) | Level 3 Interface (Data Flow Name) | Description | Interface Elements | Mode |
|------------------------------------|-------------------------------------|---------------------------------|---------------------------------------|---------------------------------------|--------------------------------------------------------------------------------------------------------------|---------------------------|--------------|
| SYM_EvtDistributeEvents (3.6.3) | SYM_DistributeStateData SD (3.2) | TBD | SYM_EvtDsdFilteredEvent | SYM_EvtDsdFilteredEvent | Event Message Object | • Event Message Object | Real Time |
| SYM_DTF Interface | | | | | | | |
| SYM_DetectFaults (3.3) | (3.6)SYM_ManageEvents | SYM_EvtHandleEvents (3.6.1) | SYM_DtfEvent | SYM_DtfEvent | Incomplete event message | • Event Message Object | Real Time |
| SYM_EvtDistributeEvents (3.6.3) | SYM_DetectFaults (3.3) | TBD | SYM_EvtDtfFilteredEvent | SYM_EvtDtfFilteredEvent | Event Message Object | • Event Message Object | Real Time |
| SYM_REF Interface | | | | | | | |
| SYM_RespondtoEvents&Faults (3.4) | (3.6)SYM_ManageEvents | SYM_EvtHandleEvents (3.6.1) | SYM_RefEvent | SYM_RefEvent | Incomplete event message | • Event Message Object | Real Time |
| SYM_EvtDistributeEvents(3.6.3) | SYM_RespondtoEvents&Faults (3.4) | TBD | SYM_EvtRefFilteredEvent | SYM_EvtRefFilteredEvent | Event Message Object | • Event Message Object | Real Time |
| SYM_ANT Interface | | | | | | | |
| SYM_PerformAnalysis (3.5) | (3.6)SYM_ManageEvents | SYM_EvtHandleEvents (3.6.1) | SYM_AntEvent | SYM_AntEvent | Incomplete event message | • Event Message Object | Real Time |
| SYM_MMD Interface | | | | | | | |
| SYM_ManageMonitorData (3.7) | (3.6)SYM_ManageEvents | SYM_EvtHandleEvents (3.6.1) | SYM_MmdEvent | SYM_MmdEvent | Incomplete event message | • Event Message Object | Real Time |
| SYM_EvtLogEvents (3.6.2) | SYM_ManageMonitorData (3.7) | TBD | SYM_EvtDataRqst | SYM_EvtLogEventRqst | Send DMG event message to archive | • Event Message Object | Real Time |
| SYM_EvtDistributeEvents (3.6.3) | SYM_ManageMonitorData(3.7) | TBD | SYM_EvtEventDBRqst | SYM_EvtEventDBRqst | Request to get information from the Event Lookup Table based on theID_Number of an Event Message | TBD | Real Time |
| SYM_ManageMonitorData(3.7) | (3.6)SYM_ManageEvents | SYM_EvtDistributeEvents (3.6.3) | SYM_EvtEventDBResp | SYM_EvtEventDBResp | Information from the event lookup Table | • Severity • Type | Real Time |

Questions or comments please contact Douglass George in Rm 35

E-mail: dgeorge@v2kmail.gsfc.nasa.gov

Phone: 918-7488

| Source | Level 2 Destination | Level 3 Destination | Level 2 Interface (Data Flow Name) | Level 3 Interface (Data Flow Name) | Description | Interface Elements | Mode |
|------------------------------------|--------------------------------|--------------------------------|---------------------------------------|---------------------------------------|--------------------------|------------------------|-----------|
| DMG Interface | | | | | | | |
| Data Management (4.0) | (3.6)SYM_ManageEvents | SYM_EvtHandleEvents (3.6.1) | G_DMG_Events | G_DMG_Events | Incomplete event message | • Event Message Object | Real Time |
| CCM Interface | | | | | | | |
| CCS Management (5.0) | (3.6)SYM_ManageEvents | SYM_EvtHandleEvents (3.6.1) | G_CCM_Events | G_CCM_Events | Incomplete event message | • Event Message Object | Real Time |
| SYM_EvtDistributeEvents (3.6.3) | CCSManagement(5.0) | | G_System_Event | G_System_Event | Event Message Object | • Event Message Object | Real Time |
| GUI Interface | | | | | | | |
| Graphical User Interface (6.0) | (3.6)SYM_ManageEvents | SYM_EvtHandleEvents (3.6.1) | G_GUI_Events | G_GUI_Events | Incomplete event message | • Event Message Object | Real Time |
| SYM_EvtDistributeEvents (3.6.3) | Graphical User Interface (3.6) | TBD | G_GUI_EvtFilteredDisplayEvent | G_GUI_EvtFilteredDisplayEvent | Event Message Object | • Event Message Object | Real Time |

5.0 Object Models & Event Trace Diagrams

The following Object Model diagrams are attached at the end of this document:

- Analysis for Manage Events
- Design for SYM_EvtAPI
- Design for SYM_ManageEvents

The following Event Trace diagrams are attached at the end of this document:

- SYM_EvtAPI Event Trace Diagram
- SYM_ManageEvents Event Trace Diagram

6.0 Class Descriptions

This section describes the classes within the Manage Events Process and the corresponding EVT API. Descriptions pertaining to the classes, class attributes, and class methods are provided.

Class attributes are provided, including the data type and a short description of each attributes purpose. Some attributes are enumerated data types. Descriptions of these enumerated data types are provided in the tables following the class descriptions. Enumerated data types will increase performance across the network and decrease maintenance.

Class methods are provided, including the parameters and return values.

6.1 Attributes and Methods

DMG_EventProxy

Allows the Event Manager class to log event messages.

Allows the Event Message class to finish building by getting the event type, event severity, and event background from the DMG database.

This class is provided and maintained by the DMG Team

RWCollectable

A Rogue Wave supplied class.

RWCollectableString

A Rogue Wave supplied class.

RWCollectableInt

A Rogue Wave supplied class.

RWSlistCollectable

A Rogue Wave supplied class.

SYM_CollectableDouble

A double

myDouble

Class Attribute

A double

SYM_CollectableFloat

A float

myFloat

Class Attribute

A float

value

Method

Returns the value of myFloat

value

Method

Returns the current value of myFloat then sets it to the float parameter passed into this method

SYM_CollectableDouble

A double

value

Method

Returns the value of myDouble

Describes the processor identification (PID) number of the CCS application that created

this event. This is used to discriminate between two identical applications running on the same node.

myProcNameInt Class Attribute

Describes the name of the CCS application that requested this event (e.g., SYM_FaultDetection)

myProcNameText Class Attribute

Describes the name of the CCS application that requested this event (e.g., SYM_FaultDetection)

mySeverity Class Attribute

Enumerated Severity levels. There are three operational (informational, alarm, and fatal) and one non-operational (debug) severity levels.

mySubSystem Class Attribute

This is an enumeration of the various subsystems of the CCS application that requested this object. The following six Subsystems currently exist within CCS: FEP, CMD, SYM, DMG, CCM, and GUI.

myTimeTag Class Attribute

Used as primary key for historical requests (along with ID Number) from DMG. Denotes the time at which the CCS application created this event message.

myType Class Attribute

Enumerated type describing the Type of this object. Examples include Telemetry and NCC types.

convertToText Method

Returns an ASCII text string of specified attributes of this object. The attributes in the text string are determined by the GUI team. This method is also used in debug mode to write to standard output.

createForeground Method
Creates the foreground
Parameter List by passing in an
existing Foreground Parameter
List (created by SYM_
EvtGenerator).

finishBuilding Method
Queries the Event Lookup table
and completes this object. The
myType, mySeverity,
myBackground, myProcNameInt
are populated.

getAll Method
Used by DMG to access private
attributes. Returns all the
attributes as a structure to DMG.

getOpMode Method
Gets the operational mode
attribute of this object.

SYM_EvtGenerator

This class will be given to every CCS application that
needs to generate event messages. CCS applications
will generate an event message. This object will then
send the event message to the EVT process.

myEventMessage Class Attribute
The Current Event message
being created.

myForeground Class Attribute
These are the foreground
parameters passed by the CCS
application.

myNetMode Class Attribute
Describes the physical
architectural LAN where this
application resides (e.g., Core,
Backbone).

myNode Class Attribute
Describes the node where the
CCS application resides that
requested this object (e.g.,
v2kma13)

myOpMode Class Attribute
Describes the operational mode
of this object. The current
proposed operational modes are
real time and historical playback.
This determines whether this
object is logged by DMG.

| | |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| myPID | Class Attribute |
| Describes the processor identification number (PID) of the CCS application that created this event. This is used to discriminate between two identical applications running on the same node. | |
| myProcName | Class Attribute |
| Describes the name of the CCS application that requested this event (e.g., SYM_FaultDetection). | |
| mySubSystem | Class Attribute |
| This is an enumeration of the various subsystems of the CCS application that requested this object. The following six Subsystems currently exist within CCS: FEP, CMD, SYM, DMG, CCM, and GUI. | |
| createAndSendEvent | Method |
| Called by CCS applications that are requesting to generate event message. This method then calls the instantiator for SYM_EvtEventMessageClass and passes the 4 source elements as parameters. | |

SYM_EvtManager

This object routes the event message object to two other processes that provide independent functionality. Real-time mode messages are sent to the log event process and to the distribute event process. Historical mode messages are only routed to the distributor process. A Queue is maintained in case the network goes down.

| | |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|
| manageEvents | Method |
| Main routine for event message processing. This routine will poll the Middleware and wait for a new event message to arrive. This method will then send the event message to the GUI subsystem. This method will also tell real time event messages to finish building themselves. The event message will send real time | |

event messages to the DMG for logging.

SYM_EvtMiddleware

This class is provided by the Middleware Team

SYM_EvtProxy

Proxy for Internal Communication among SYM processes

The following table provides the types of information an event message object will encapsulate and distribute around CCS.

Table3 Event Message Size

| Attribute | Data Type | Bits |
|---------------------------------------------------------|-------------|------------|
| ID_Number | Short | 16 |
| CCS_Time | Double | 64 |
| Type | Short | 16 |
| Severity | Short | 16 |
| Operational Mode | Short | 16 |
| Source_Subsystem | Short | 16 |
| Source_Node | Short | 16 |
| Source_ProcessName | Short | 16 |
| Source_PID | Integer | 32 |
| Foreground Data | Linked List | 0 minimum |
| Minimum Size | | 208 |
| Average Size (Foreground = 4 float elements) | | 336 |

The following tables provide descriptions for the attributes that are of enumerated type.

Table4 Event MessageSource_Subsystem Enumeration

| Enumeration | Flag Name | Description |
|-------------|-----------|---------------------|
| 0 | Reserved | Reserved |
| 1 | FEP | Front End Processor |
| 2 | CMD | Command Processor |

Questions or comments please contact Douglass George in Rm 35

E-mail: dgeorge@v2kmail.gsfc.nasa.gov

Phone: 918-7488

| Enumeration | Flag Name | Description |
|-------------|-----------|--------------------------|
| 3 | SYM | System Monitoring |
| 4 | DMG | Data Management |
| 5 | CCM | CCS Management |
| 6 | GUI | Graphical User Interface |

Table 5 Event Message Type Enumeration

| Enumeration | Flag Name |
|-------------|------------------------|
| 0 | Reserved |
| 1 | Alarm |
| 2 | Command |
| 3 | Configuration |
| 4 | NCC |
| 5 | Telemetry |
| 6 | PSTOL Response |
| 7 | PSTOL Input |
| 8 | System/Software Events |
| 9 | General Events |
| 10 | PSTOL echo |
| 11 | Dialog |
| 12 | File to File |
| 13 | Command Page |
| 14 | Telemetry |
| 15 | Playback |
| 16 | Real-Time |
| 17 - 65,536 | Reserved |

Note: Numbers 1-13 are reserved for PRS legacy event message types.

Table 6 Event Message Severity Enumeration

| Enumeration | Flag Name | Description |
|-------------|---------------|----------------------------------------|
| 0 | Debug | Not Logged. For testing purposes only. |
| 1 | Informational | No action needed |

Questions or comments please contact Douglass George in Rm 35

E-mail: dgeorge@v2kmail.gsfc.nasa.gov

Phone: 918-7488

| Enumeration | Flag Name | Description |
|-------------|-----------|-------------------------------|
| 2 | Error | Action needed - Low Priority |
| 3 | Warning | Action needed - High Priority |

Questions or comments please contact Douglass George in Rm 35

E-mail: dgeorge@v2kmail.gsfc.nasa.gov

Phone: 918-7488

7.0 Future Capabilities

The following functionality will be provided in future releases:

- Filtering and Distributing event messages to CCM and specific SYM processes.